An Introduction to Boost

Andrew Walker, 6 Jul 2003

4.69 (35 votes)

An overview of the Boost library

Contents

Introduction

Boost is a free library which is aimed at providing quality software components to developers, whilst using the styles of the Standard Template Library. Some of the components within the library may be put forward as future extensions to the Standard Library.

The Boost main page
The Boost libraries index
CUJ Descriptions of the boost components

The Boost homepage contains extensive documentation on all of the individual components. This article is intended to be an overview of why you should consider using Boost, and the Boost components in preference to other libraries, and to provide a location on CodeProject which can offer links to related documentation.

Installation

Boost installation is simple because most of the components within the library reside in their own header files, which should not require modification. The BoostJam build tool is available for the components which do require compilation.

You can download the source zip from SourceForge.

Unzip the entire archive into a directory of your choosing. To start using the components add the Boost directory which includes version numbers to your include path. For the current version of Boost, boost_1_30_0.

Most of the components come with test suites and examples of use.

Why Boost?

- Boost is namespace aware. All components within the library are packaged in the "boost" namespace, or a sub-namespace thereof.
- Regular updates. Boost is a library which is growing all the time, the home page and the Boost Announcements Lists show some of the changes in the last few releases.
- Developer support. Questions related to the components can be directed to the Boost users mailing group, or found on the Boost Mail Archives or on one of the specific lists for a particular component.
- Boost supports a variety of compilers, operating systems and standard libraries. It provides workarounds for the broken features of many compilers, perhaps the most significant being the workarounds for problems with templates, including partial template specialisation and member template friends.
- Regression testing. Each update to Boost is heavily regression tested, and the status of the library for all compilers is freely available.
- Many of the people involved with the development of the C++ standard are involved with Boost.
- Simple to install and upgrade. In most cases, installation and upgrading only requires the addition or change of one include path.
- Easy to configure. Compilation options can be specified by changing directives in one or two header files.

What Does Boost Provide?

What follows is a minimal listing of components. There are about 50 major sub-components in Boost at the moment. The following components were those that I felt logically progressed from the components in the STL, were easy to integrate, or were especially significant to most programmers.

Smart Pointers

Smart pointers are tools that prevent resource leaks (especially in the presence of exceptions), promote the concept of 'Initialisation is Resource Acquisition'. They emulate, to a certain extent, garbage collection like behaviour.

Most of the limitation of std::auto_ptr are relatively well known:

- `std::auto_ptr`'s cannot be stored within a standard containers.
- `std::auto_ptr`'s cannot (easily) be used to implement the pImpl (pointer-to-implementation) idiom.
- `std::auto_ptr` does not work with arrays.

The 5 types of Boost smart pointers overcome these flaws and provide many extra features.

- Custom delete functions can be supplied.
- Detection of incomplete template types.

Boost smart pointer index
Comparison of Boost and Loki Smart Pointers The New C++: Smart(er) Pointers - Herb Sutter
Introduction to uses of the Boost smart pointers Conversations: Getting to the Point - Herb Sutter and Jim Hyslop

Composers

Functors and binders have become a common part of using the STL, but using most standard library implementations it can still be difficult to combine multiple functions. Composers allow functors to be combined in several ways, minimising the amount of times that users have to write their own loops.

Boost::compose index
The C++ Standard Library - A Tutorial and Reference (Nicolai M. Josuttis)

Any

A component which provides a type safe way to move any type of component, without having to rely upon void pointers or unions. The design principles for this component is at least as significant as the component itself (derivation of a template class from a non template base class). Something similar to `boost::any` appears in Alexandrescu's Modern C++ Design in the guise of Functors and Functor Implementations.

boost::any main page
boost::any Theory
Introduction to uses of Boost Any Conversations: I'd Hold Anything for You - Herb Sutter

Bind and Function

Bind and Function are specified as two separate components, but they are extensions (any number of arguments) of binders and functors concept which are currently in place in the Standard Library.

boost::function main page
boost::bind main page

The Lambda Library

The lambda library provides a shortcut for producing binders, functors and composers using expression templates. My personal opinion on the library is that developers would need some practice to recognise it's use. Libraries like the Lambda library are probably the way of the future for C++, but at the moment, I think I'm prepared to have slower uglier code that I know the next guy can understand.

The Boost Lambda Library Index
Further information on the basics of expression templates was published in the March issue of the C/C++ users journal (C++ Expression Templates – Angelika Langer and Klaus Kreft)

The Boost Graph Library (The BGL)

The BGL is a huge library, with a large amount of support material and good sample programs. "The Boost Graph Library, The: User Guide and Reference Manual" has been published by Addison-Wesley in the C++ In Depth Series (The same fantastic series that includes 'Exceptional C++', 'More Exceptional C++' and 'Modern C++ Design'), which I believe is testament to the quality of the library.

Boost Graph Library Table of Contents
The Boost Graph Library, The: User Guide and Reference Manual Book Review

Thread

Developed by Code Project regular William E. Kempf, the threads library makes it seem almost as easy to do threads in C++ as

it is in Java. It requires linking to an additional library, built with BoostJam.

boost::threads index
William E. Kempf on boost::thread

Spirit parser generator framework

Jonathan de Halleux has written an excellent introduction to the Spirit Parser Generator Framework with comprehensive links to relevant material

What Else?

- Regular Expressions
- Traits
- File System (Directory Iteration)
- Iterator Adaptors
- Maths and Matrices
- A Template metaprogramming framework
- Tuples
- Python

References

Many of the references in this article come from the C / C++ Users journal website, which is an excellent resource for up to date information on uses, and techniques for using STL and Boost.

Additional information about Boost can be found at Boost Consulting

This article was inspired by an item in the 'Article Requests and Ideas' Forum by John M. Drescher

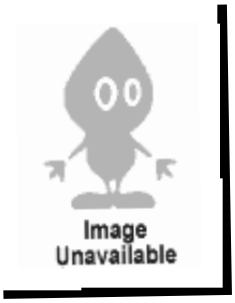
Version history

30. 6. 2003 Initial Posting

License

Share

About the Author



Andrew Walker

Web Developer

Australia

Andrew is a PhD student at Swinburne University in Melbourne Australia, investigating the control systems of UUV's - Unmanned Underwater Vehicles. He graduated from Swinburne with a Bachelor of Engineering (Robotics and Mechatronics) and a Bachelor of Science(Computer Science & Software Engineering)

His practical experience includes a year developing an industrial computer vision system from scratch, and working as the software architect for the 2004 Swinburne Robocup team (f180 league).

Comments and Discussions

33 messages have been posted for this article Visit **http://www.codeproject.com/Articles/4496/An-Introduction-to-Boost** to post and view comments on this article, or click **here** to get a print view with messages.